Descubrimiento dinámico de recursos en sistemas de objetos distribuidos

Ariel Fuxman **Pablo R. Fillottrani**

Departamento de Ciencias de la Computación
Universidad Nacional del Sur
Av. Alem 1253 – (8000) Bahía Blanca
Argentina

e-mail: afuxman@acm.org, ccfillo@criba.edu.ar

Resumen

En los sistemas orientados a objetos tradicionales, el acceso a los proveedores de servicio se lleva a cabo de manera sencilla a partir de referencias explícitas, tales como punteros o nombres. Sin embargo, la situación es mucho más complicada en los *sistemas de objetos distribuidos*, debido a su gran complejidad. En este trabajo se estudia en detalle este problema y se plantean alternativas de solución a través de mecanismos de *descubrimiento dinámico de recursos*. Fundamentalmente, se describe un mecanismo de descubrimiento dinámico denominado *Trading* en una presentación que sigue el estilo de los patrones de Gamma *et al* [Gamma 95] y se trata una serie de consideraciones importantes tales como la Federación de Traders, la representación de las ofertas de servicio y aspectos de correctitud y seguridad.

1 Introducción.

Los *sistemas de objetos distribuidos* se basan en la aplicación de los principios de la *orientación a objetos* al diseño e implementación de *sistemas distribuidos*. Su objetivo es permitir que el usuario pueda acceder a los distintos componentes del sistema con total abstracción de los detalles de implementación (por el encapsulamiento, propio de la orientación a objetos) y su ubicación física (por la transparencia de locación, propia de los sistemas distribuidos). Estos sistemas son por lo general muy complejos debido a su escala ilimitada (existen sistemas que cubren todo el planeta) y su dinamismo (los objetos y servicios disponibles evolucionan continuamente).

En todo sistema orientado a objetos, los objetos asumen los roles de *clientes* y *proveedores* de servicio. Los proveedores brindan servicios (operaciones) que pueden ser invocados por los demás objetos del sistema, que actúan como clientes. En los sistemas orientados a objetos tradicionales, el acceso a los proveedores se logra mediante el uso de referencias explícitas (punteros o nombres) que deben ser conocidas antes de invocar cualquier operación. Ésto resulta mucho más difícil en los sistemas de objetos distribuidos, debido a su gran complejidad. Por un lado, dada la gran escala de estos sistemas, la cantidad de servicios disponibles hace imposible que el desarrollador pueda recordar sus nombres y, menos aún, su comportamiento. Por otra parte, los objetos en ejecución están constantemente cambiando de ubicación (migrando) y modificando dinámicamente la semántica de los servicios que ofrecen. En este contexto las referencias explícitas ya no son apropiadas porque, por lo general, cuando se necesita un servicio, resulta imposible disponer de una referencia a su proveedor. Esta limitación impulsó la exploración de mecanismos encargados de la vinculación dinámica entre objetos, que permitan ubicar, o *descubrir*, al proveedor apropiado en cada caso sin necesidad de tener una referencia explícita. Esta vinculación se logra a través de mecanismos conocidos como de *descubrimiento dinámico de recursos*.

Un mecanismo muy importante para el descubrimiento dinámico es el que se denomina *Trading*. Su objetivo es permitir que los clientes descubran al proveedor de servicio a partir de la especificación de las características del servicio que ese proveedor debe brindar. Para ésto, se incorpora al sistema un componente denominado *Trader*, que actúa como mediador entre los clientes y los proveedores de servicio. Los proveedores deben registrar sus servicios en el Trader mediante *ofertas de servicio*. Los potenciales clientes informan al Trader las características del servicio que requieren. El Trader intenta hacer corresponder el requerimiento del cliente con las ofertas registradas. Si es posible realizar esta correspondencia, el cliente recibe una referencia que le permite interactuar directamente con el proveedor del servicio.

En este trabajo se presenta al mecanismo de Trading a través de un patrón de diseño y se discuten una serie de consideraciones importantes tales como la Federación de Traders, la representación de las ofertas de servicio y aspectos de correctitud y seguridad. La organización del trabajo sigue los lineamientos de los patrones de diseño, con secciones dedicadas a los objetivos, motivación, aplicabilidad, estructura, participantes, colaboraciones, consecuencias y aplicaciones del mecanismo de Trading.

2 El Patrón TRADER

El mecanismo de Trading constituye una solución de diseño al problema del descubrimiento de recursos en ambientes muy complejos, como los sistemas de objetos distribuidos. Por lo tanto, resulta interesante presentarlo como un *patrón de diseño*.En este trabajo, presentaremos un patrón al que denominaremos TRADER, siguiendo el estilo de los patrones de Gamma *et al* [Gamma 95].

TRADER

Clasificación : *Comportamiento* de *Objetos*.

Objetivo

Permitir la invocación de servicios sin la necesidad de tener referencias explícitas a sus proveedores.

Motivación

En todo sistema orientado a objetos, la invocación de operaciones (*servicios*) se realiza sobre objetos denominados *proveedores de servicio*. La manera tradicional de acceder a esos proveedores es a través de referencias explícitas (punteros, nombres) que deben conocerse antes de la invocación.

En los sistemas de objetos distribuidos la situación es considerablemente más complicada porque, debido a su gran escala y complejidad, es imposible que el cliente que necesita un servicio disponga de referencias a un proveedor apropiado. Por lo tanto, las referencias explícitas no resultan suficientes y surge, por ende, la necesidad de un mecanismo que permita *descubrir* en ejecución los objetos que están en condiciones de llevar a cabo el servicio que se desea invocar.

Como ejemplo, tomemos el caso de un sistema que integra un conjunto de instituciones bancarias. Supongamos que un usuario pretende abrir una caja de ahorros pero no conoce ninguno de los bancos participantes. Una solución a este problema puede ser la incorporación al sistema de un nuevo componente al que denominaremos Trader. En el Trader, los bancos registrarán los servicios que ofrecen, como por ejemplo apertura de cuentas bancarias (especificando qué tipo de cuenta), transacciones bancarias (depósitos, extracciones de dinero, etc.), transferencias a otros bancos, etc. Entonces, lo que el usuario deberá hacer es solicitar al Trader que le informe todos los bancos en los que se puede abrir la caja de ahorro que cumple con sus necesidades. El escenario resultante sería el siguiente:

1. Los bancos asociados al sistema (*exportadores*) registran en el Trader *ofertas de servicio* especificando los *tipos de servicio* que ofrecen (apertura de una cuenta, por ejemplo) junto con sus *propiedades de servicio* tales como, por ejemplo, tipo de cuenta (caja de ahorro, cuenta corriente, plazo fijo, etc.), interés mensual, costo de mantenimiento, posibilidad de operar con cajero automático, etc.

2. El usuario informa al Trader el servicio que necesita, por ejemplo apertura de cuenta en caja de ahorros con un interés mensual superior al 1% y con

posibilidad de operar en cajero automático.

3. El Trader busca entre todas las ofertas de servicio registradas y entrega al usuario referencias a bancos en los que puede abrir el tipo de cuenta deseado.

4. El usuario evalúa los servicios que ofrecen los bancos que le informó el Trader, selecciona uno de ellos y le solicita el servicio de apertura de una caja de ahorros.

Aplicabilidad

Es conveniente usar el patrón TRADER cuando

- se pretende desacoplar a los clientes de los servicios que solicitan.
- se necesita que los servicios estén ampliamente disponibles para la mayor cantidad posible de objetos del sistema.

Estructura

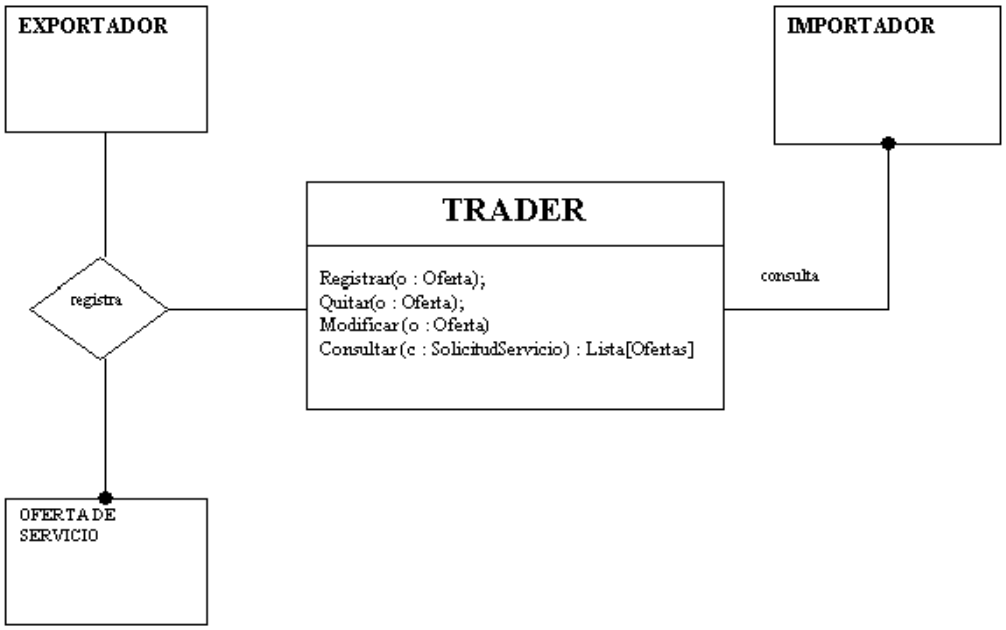

Figura 1: Estructura del patrón TRADER.

Participantes

- Trader : componente del sistema que se encarga de registrar las ofertas de los exportadores y de encontrar las ofertas que satisfacen las necesidades de los importadores. Brinda operaciones para que los exportadores registren, quiten y modifiquen ofertas y para que los importadores puedan realizar consultas.

- Exportadores : ofrecen al resto del sistema los servicios que disponen (ej : los bancos) a través de ofertas de servicio.
- Ofertas de servicio : especifican las características del servicio ofrecido (ej. : tipo de cuenta, posiblidad de operar con cajero, etc.) y mantienen una referencia explícita al exportador del servicio.
- Importadores : necesitan invocar un servicio determinado pero desconocen quién se los puede proveer (ej : usuarios del sistema bancario). Pueden realizar consultas al Trader del sistema.

Colaboraciones

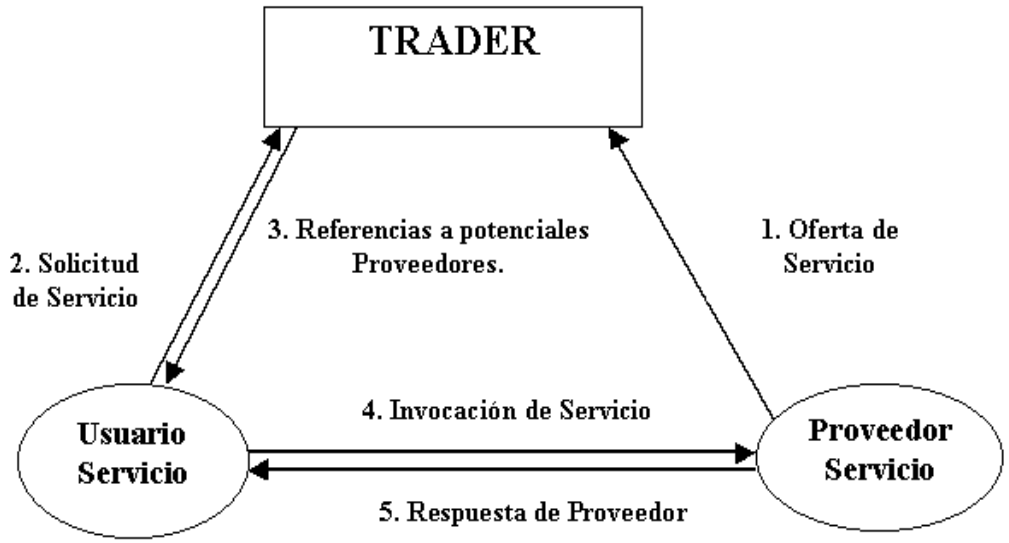

Figura 2: Mecanismo de Trading

En la figura 2 se muestra la interacción entre los participantes del patrón, que involucra los siguientes pasos :

1. Oferta de servicio : Los exportadores publican sus servicios en el Trader a través de ofertas de servicio
2. Solicitud de servicio : Un cliente necesita un servicio determinado, pero no tiene referencias explícitas a ningún proveedor adecuado. Entonces, lo que hace es consultar al Trader, entregándole una *solicitud de servicio*, que especifica las características del servicio requerido. Debe notarse que para contactar al Trader se necesita una referencia explícita (que podría ser la única que se necesita inicialmente en el sistema) u obtener la referencia a través de otro Trader (que estaría actuando como *metaTrader*).

3. Correspondencia entre ofertas y solicitudes de servicio : El Trader busca entre las ofertas registradas y devuelve referencias explícitas a proveedores que cumplen con los requerimientos de la solicitud de servicio

4. Invocación de servicio y respuesta del proveedor : Una vez que dispone de referencias explícitas, el cliente puede utilizarlas para invocar el servicio deseado sobre el proveedor y obtener una respuesta.

Consecuencias

El patrón TRADER tiene las siguientes ventajas y desventajas :

- *Reduce el acoplamiento entre los proveedores de servicio y sus clientes.* En el diseño orientado a objetos es usual que los proveedores de servicio no realicen suposiciones acerca de los clientes (excepto en los casos de call-back). Sin embargo, el cliente siempre conoce, en mayor o menor medida, la interface que debe satisfacer el proveedor. Esto se puede evitar mediante el patrón TRADER ya que el cliente no debe asumir nada acerca del proveedor del servicio, el cual será contactado solo a través del Trader.

- *Favorece la amplia difusión de los servicios de un proveedor.* Se puede decir que el Trader expande el "mercado" de potenciales consumidores de un servicio a todos los objetos del sistema. Si no existiera el Trader, los servicios solo serían utilizados por el subconjunto de objetos del sistema desarrollados por programadores que tenían noción de su existencia.

- *Facilita la replicación de servicios.* Como los clientes sólo hacen referencia a las características del servicio solicitado y es el Trader quien ser encarga de realizar la correspondencia, es posible que distintas instancias de una misma clase se contacten con proveedores de servicio diferentes para acceder al mismo servicio.

- *Aumenta el costo computacional de la vinculación entre el cliente y el proveedor del servicio.* La búsqueda de las ofertas que satisfacen el requerimiento es de orden lineal respecto al número de ofertas registradas frente al orden constante que implica una invocación si se dispone de referencias explícitas.

- *Se crea una gran dependencia de un componente central.* Todos los objetos deberán acceder al Trader para encontrar a los proveedores de servicio. Si el sistema crece, aumentará la cantidad de requerimientos y el número de ofertas a registrar, por lo que puede plantearse un problema de escalabilidad. Este problema, entre otros, ha llevado a la búsqueda de soluciones tales como la Federación de Traders.

Aplicaciones

- La estandarización más difundida del concepto de Trading es la Función de Trading [ITUISO 97] definida por la ODP (Open Distributed Processing). ODP brinda una especificación general para el desarrollo de entornos para

aplicaciones abiertas y distribuidas. El modelo básico de referencia de ODP (RM-ODP) [ODP1 96, ODP2 95, ODP3 95] es un estándard internacional desarrollado por la ISO y la CCITT. La Función de Trading, identificada y posicionada en el Modelo de Referencia RM-ODP, ha inspirado la mayoría de las especificaciones e implementaciones de Trading que se han desarrollado hasta el momento, incluyendo al *Trading Service* de CORBA definido en [OMG 96].

El estándard plantea una serie de políticas que deben estar presentes en toda implementación, tales como políticas de búsqueda, de consumición de recursos, de seguridad, de manejo de recursos, etc. Las ofertas de servicio son instancias de un tipo de servicio, que consiste en una tupla de propiedades de servicio y contienen una referencia a la interface del proveedor del servicio. La especificación consta de cinco interfaces: Lookup, Register, Admin, Link y Proxy. La interface Lookup brinda la operaciones que permiten que los importadores realicen solicitudes al Trader. La interface Register brinda operaciones para que los exportadores registren sus ofertas en el Trader y para remover o alterar esas ofertas. La interface Admin provee operaciones para listar las ofertas registradas en el Trader. La interface Link brinda operaciones para manejar los *vínculos* entre Traders federados en un grafo de Federación. Finalmente, la interface Proxy permite manejar ofertas proxy, que son una forma especial de oferta de servicio que, en vez de contener una referencia a la interface del proveedor de servicio contiene una referencia a una interface que a su vez se encarga de devolver una referencia al proveedor, brindando así un nivel más de indirección. Las ofertas proxy pueden ser usadas para soportar el encapsulamiento de sistemas *legacy* o como una factoría de servicios que los crea en el momento en que son requeridos.

- Se puede utilizar el patrón para implementar un sistema de "páginas amarillas" en el que el usuario interactúe con el Trader para obtener servicios basándose en su categoría. Por ejemplo, el usuario podría solicitar información sobre bancos que brindan una tasa de interés determinada. El Trader devolvería referencias para operar con esos bancos, que el usuario utilizará en la manera que considere apropiada.

- En la Washington University in Saint Louis se combinó el servicio de Trading de CORBA con un servicio para provisión de servicios multimedia [Widdof 98]. El sistema permite seleccionar aplicaciones multimediales especificando el título y las propiedades de calidad de servicio (QoS) deseadas tales como ancho de banda, provisión continua, etc.

- En un dominio de aplicación totalmente diferente, Janowski et al [Janowski 96] formalizaron el concepto de Trading en el lenguaje formal RSL (Raise Specification Language) y lo utilizaron para desarrollar un modelo formal para empresas que interactúan entre sí, colaborando o compitiendo.

Patrones relacionados

Al igual que en el patrón MEDIATOR definido por [Gamma 95], se reduce el acoplamiento entre los objetos ya que no deben conocerse entre sí e interactúan a través de un componente central (el mediador o el Trader). La diferencia fundamental es que el mediador conoce todas las interacciones posibles entre los objetos y las

lleva a cabo en nombre de ellos. En cambio el TRADER solo establece el contacto inicial entre los objetos que van a interactuar. A partir de ahí la comunicación es directa entre el cliente y el proveedor de servicio.

Implementación

Los siguientes son algunos aspectos importantes a considerar acerca del mecanismo de Trading :

- *Mecanismo de invocación.* El cliente no sólo no tiene referencias explícitas al proveedor de servicio sino que tampoco conoce el tipo de su interface. Más aún, las distintas ofertas que le devuelve el Trader pueden hacer referencia a proveedores con distinto tipo de interface. Ésto impide que las llamadas puedan realizarse mediante un mecanismo estático, que se resuelva en tiempo de compilación. Por lo tanto, el sistema en que se utilice este patrón deberá permitir mecanismos de *invocación dinámica*, mediante los cuales se pueda recuperar en ejecución la interface de un objeto conociendo su tipo y, a partir de ella, realizar la invocación a alguno de sus servicios. En el caso de CORBA, esta funcionalidad se logra a través del Interface Repository (un repositorio en el que se almacenan las especificaciones de las interfaces) y su mecanismo de invocación dinámico denominado Dynamic Invocation Interface (DII).

- *Federación de Traders.* La gran dependencia que tiene este patrón del Trader como componente central atenta contra su escalabilidad. Una solución posible consiste en particionar el espacio de ofertas. Cada partición del espacio de búsqueda, a la que llamaremos *dominio* será atendida por un Trader. Cada Trader estará *vinculado* con otros Trader con los cuales podrá comunicarse. De esta forma se conforma un *grafo de Trading* donde cada Trader y su dominio de ofertas serán los nodos. A pesar de estar vinculados, cada Trader mantendrá totalmente su autonomía, pudiendo aceptar o rechazar las consultas que le realizan Traders de otros dominios. A esta configuración de Traders autónomos vinculados entre sí la llamaremos *Federación de Traders*.

  Cuando un Trader no pueda resolver una solicitud de servicio o cuando necesite buscar una mayor cantidad de ofertas para devolver como respuesta deberá *propagar* el requerimiento hacia los Traders con los que está vinculado en el grafo de Trading. En los vínculos se mantendrán *políticas de propagación* que establecerán cuándo se permite que un requerimiento sea enviado al Trader siguiente. La misma situación puede ocurrir con la registración de las ofertas. Si un Trader no dispone de espacio de almacenamiento para una oferta, podrá propagarla para que sea registrada en algún otro Trader del grafo.

  Una ventaja adicional de la Federación es que cada Trader podrá adaptarse a las características particulares del dominio en el que opera, manteniendo políticas e implementaciones diferentes. Los vínculos entre dominios serán los encargados de enmascarar estas diferencias, mapeando los requerimientos y resultados para que sean entendidos por los Trader que se comunican.

  Esta distribución del espacio de ofertas debe ser totalmente transparente para el usuario. Los exportadores se comunicarán con un solo Trader para registrar sus ofertas aunque quizá sean almacenadas en algún otro dominio. Los importadores se contactarán con un único Trader y éste será el encargado de explorar el grafo de Trading para encontrar ofertas en otros Trader si es necesario.

- *Especificación de las ofertas de servicio.* Es claro que la especificación de las ofertas de servicio tiene una profunda influencia en la manera en que se realizarán las consultas y en el mecanismo de correspondencia entre consultas y ofertas. La especificación de las ofertas debe ser tal que no dificulte el desarrollo de un mecanismo para detectar las consultas que se corresponden con las ofertas que el Trader tiene registradas. Una notación para la especificación de ofertas debe estar basada en una *sintaxis precisamente definida* para evitar ambiguedades. Este requerimiento se origina en el hecho de que el Trader debe tener bases sólidas para el algoritmo de correspondencia e implica la necesidad de una *definición explícita*. Un segundo requerimiento es que la notación debe ser suficientemente *abierta* para evitar la necesidad de una estandarización a priori de las descripciones de servicios. Este requerimiento es de tipo pragmático e implica que si el sistema será abierto, permitiendo la incorporación de nuevos servicios no previstos por el usuario, entonces no se puede obligar a tener un conocimiento a priori de las descripciones de todos los servicios. Esto conduce -en contraposición al primer requerimiento- a una *definición implícita* de la descripción del servicio. La solución de esta aparente contradicción para la especificación de ofertas es uno de los desafíos más importantes en la investigación de los mecanismos para el descubrimiento dinámico de recursos. Finalmente, la notación debe capturar en la mayor medida posible la semántica de los servicios que especifica.

  Hasta el momento, todas las aproximaciones para la especificación de ofertas han utilizado un sistema de tipos. La alternativa más simple es utilizar durante la ejecución la información estática que se posee acerca de la interface de los objetos del sistema. El tipo de una oferta de servicio sería la signatura de la interface del proveedor del servicio y en base a ella podrían realizarse las consultas. Claramente, al utilizarse únicamente la información sintáctica que se posee acerca del servicio, esta alternativa no modela el comportamiento de las distintas ofertas. Otra posibilidad consiste en asignar a cada oferta de servicio un tipo independiente del tipo de la interface del objeto que lleva a cabo el servicio. En la Función de Trading de ODP, por ejemplo, las ofertas son instancias de un tipo de servicio que consiste en una tupla de *propiedades de servicio*. Por ejemplo al tipo de servicio `cuentaBancaria` se le pueden asociar las propiedades `interesMensual, operaConCajeroAutomatico, otorgaChequera`, etc. Las consultas se pueden realizar especificando el tipo de servicio y un predicado cuyas variables sean las propiedades de ese tipo (por ejemplo, `cuentaBancaria : interesMensual > 10 % AND operaConCajeroAutomatico`).

  Una forma sencilla de obtener una especificación abierta consiste en aplicar los conceptos de la orientación a objetos a la especificación de los tipos de servicio, estableciendo relaciones de herencia que permitan la aplicación del polimorfismo de inclusión [Cardelli 85]. De esta manera, cuando el usuario consulta por un tipo T en realidad está dispuesto a aceptar un servicio que corresponda a cualquiera de los subtipos de T, incluso aquéllos de los que no tiene noción de su existencia. Una propuesta más ambiciosa es la de [Puder 94] que se basa en técnicas de inteligencia artificial, en particular provenientes del campo del aprendizaje computacional (machine learning). Su trabajo se basa en un método de representación del conocimiento llamado *grafos conceptuales* [Sowa 84] y permite que la representación de las ofertas sea refinada a medida

que el usuario interactúa con el Trader.

Finalmente, la alternativa más poderosa, aunque más costosa en términos computacionales, es la especificación mediante fórmulas pertenecientes a una teoría formal. De esta manera se tendrá un mecanismo de correspondencia mucho más poderoso (basado en el esquema computacional de la teoría) y un mayor poder expresivo para especificar la semántica de los servicios que representan las ofertas.

- *Verificación de la correctitud de las especificaciones.* El encargado de realizar las especificaciones de las ofertas será siempre el usuario que interactúa con el sistema o el desarrollador de aplicaciones. Es claro que, ya sea en forma accidental o deliberada, la especificación que se formula para las ofertas puede no corresponderse con su comportamiento real.

  En un ambiente basado en Trading, resulta razonable que sea el Trader, que se encarga de realizar el contacto inicial entre el cliente y el proveedor de servicio, quien se responsabilice acerca de la validez de la especificación de las ofertas que tiene registradas. Sin embargo, este problema no ha sido encarado aún en las especificaciones e implementaciones de Trading. Por ejemplo, el estándard de la Función de Trading de ODP aclara explícitamente que el Trader no puede hacerse responsable por la calidad de los servicios descriptos en las ofertas de servicio.

  La alternativa más sencilla para encarar este problema consiste en asumir que existe un contrato entre el exportador del servicio y el Trader por el cual el exportador se compromete a presentar una especificación correcta. Así, el dominio al que accede cada Trader puede asumirse como confiable. Sin embargo, la incertidumbre acerca de la correctitud de las especificaciones sigue existiendo si se permite la Federación de Traders, ya que un Trader no puede garantizar nada acerca de las ofertas registradas en el dominio de otro Trader.

  Otra alternativa es que los Trader soporten un protocolo mediante el cual se pueda asignar un "grado de confiabilidad" a la especificación de los servicios. Algunos de los servicios pueden ser calificados por los administradores del Trader, quienes se encargarían de verificar su correctitud en forma manual o utilizando alguna herramienta automatizada. Ésto podría combinarse con un mecanismo mediante el cual el cliente pueda informar al Trader su grado de satisfacción con el servicio que se le entregó. De esta manera, el Trader podrá tomar una política apropiada con aquellas ofertas que, a juicio del cliente, no cumplen con la especificación de la oferta. El problema aquí es que el Trader se está basando en el juicio de los clientes, sin saber si éstos son confiables.

  Finalmente, lo ideal sería brindar mecanismos para que el Trader realice la verificación automática de las ofertas que recibe para ser registradas. Las únicas especificaciones que podrían utilizarse para una verificación automática son las basadas en teorías formales y, aún así, las herramientas de verificación automática existentes en la actualidad son muy limitadas y, por supuesto, no apuntan en absoluto a ser utilizadas "on-line" durante la ejecución del sistema.

- *Política de seguridad.*

  Ante la dificultad práctica de asegurar el correcto comportamiento de los servicios a partir de la especificación de las ofertas, puede optarse por al menos

proteger a los recursos del sistema del uso indebido o no autorizado. Ésto se logra adoptando una *política de seguridad*.

En los sistemas cliente-servidor sencillos con relativamente pocos componentes, las relaciones de confianza entre los componentes son claras. Por ejemplo, los clientes de un servidor de archivos confían en el servidor, aunque no vice versa.

En los sistemas de objetos distribuidos, en cambio, la situación es mucho más compleja y resulta mucho más difícil plantear *fronteras de confianza* dentro del sistema. Las fronteras "naturales" entre objetos provistas por el encapsulamiento no siempre son las más apropiadas, ya que no necesariamente estarán soportadas por mecanismos de seguridad. Si se usa únicamente al servicio de Trading para establecer el contacto entre objetos y obtener referencias válidas a los proveedores, entonces puede establecerse una colaboración muy productiva entre el Trader y resto de la infraestructura del sistema para el establecimiento de una arquitectura de seguridad.

La *autorización* y *control de acceso* a los servicios es la funcionalidad que mejor se puede centralizar en el Trader. Cada oferta debería especificar qué usuarios o grupos de usuarios pueden acceder al servicio que representan. Entonces, cuando un cliente realice una consulta, el Trader ignorará las ofertas referidas al servicio al que el cliente no esté autorizado a acceder. Si el Trader es el único encargado de devolver referencias válidas para acceder a los proveedores, el proveedor tiene la garantía que todos los clientes que accedan a sus servicios estarán debidamente autorizados y no tendrá que implementar su propia política de seguridad. La implementación de mecanismos de *auditoría de seguridad* para controlar a los usuarios también se simplifica, ya que puede restringirse a que el Trader se encargue de registrar los clientes que se contactan con él y las referencias que les entrega. La política de seguridad del Trader debe extenderse al caso de Federación de Traders. Cada Trader debería establecer privilegios para los Traders vecinos en el grafo y algún mecanismo adecuado para la propagación de esos privilegios.

Referencias

[Cardelli 85] Luca Cardelli y Peter Wegner, "On Understanding types, data abstraction, and polymorphism", ACM Computing Surveys, 17(4):471-522, 1985.

[Gamma 95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns : Elements of Reusable Object-Oriented Software", Addison-Wesley Proffesional Series, 1995.

[ITUISO 97] ITU/ISO, "ODP Trading Function : Specification. Draft Rec. X950-1—ISO/IEC 13235-1", 1997.

[Janowski 96] Tomasz Janowski y Cleta Milagros Acebedo, "The Market Game: Towards the Formal Model For Cooperating Enterprises", Workshop on Theoretical Problems on Manufacturing System Design and Control, Lisbon, Portugal, 1996.

[OMG 96] OMG, "OMG CORBAServices : Common Object Services Specification", OMG Document Number 97-12-23, Object Management Group, Framingham, MA, 1996.

[ODP1 96] ITU/ISO, "Reference Model of Open Distributed Processing - Part 1 : Overview", ISO/IEC 10746-1, ITU-T Rec X901, 1996.

[ODP2 95] ITU/ISO, "Reference Model of Open Distributed Processing - Part 2 : Foundations", ITU-T Rec X.902, 1995.

[ODP3 95] ITU/ISO, "Reference Model of Open Distributed Processing - Part 3 : Architecture", ITU-T Rec X903, 1995.

[Puder 94] A. Puder, S. Markwitz, F. Gudermann y K. Geihs, "AI-based Trading in Open Distributed Environments", University of Frankfurt, Germany, 1994.

[Sowa 84] John Sowa, "Conceptual Structures, information processing, mind and machine", Addison-Wesley Publishing Company, 1984.

[Widdof 98] Seth Widoff, "Using the COS Trading Service to discover Best-Matched Servers in a Distributed Multimedia Environment", Tesis de grado, Washington University in Saint Louis, 1998.